IN THE CLAIMS

Please amend Claims 1-4, 6-8, 11-12, 14, 19-21, 25, 28-32, and 36-40, and add new Claims 41-43, as follows:

1. (Currently amended) In an embedded-DRAM (dynamic random access memory) processor incorporating wide data paths to memory, a method of intelligent caching comprising the steps of:

segmenting the architecture into first and second portions;

executing instructions by said first portion which manipulate only register operands; and

executing instructions by said second portion which perform row-oriented load/store operations as well as individual register-to-register move operations;

wherein:

at a given time said first portion of said architecture sees a first subset of the total available registers as its set of architectural registers while a second subset of said total available registers is not accessible by said first portion of said architecture ;

said first portion of said architecture comprises one or more functional units which execute a first program comprising instructions using register operands; and

said second portion of said architecture executes a second program tightly coupled to said first program, said second program comprising parallel row-oriented load/store/mask commands, register-to-register move commands, and architectural register set switch commands to insure that data accessed by said first program is available when it is needed.

2. (Currently amended) For use in a system involving an Embedded-DRAM processor, a [[A]] method for intelligent caching comprising the steps of:

splitting an architecture into first and second portions, said first portion comprising a set of functional units and a set of architectural registers exercised thereby, said second portion comprising at least one functional unit capable of moving data

between a main memory implemented as one or more banks of DRAM without a caching system that employs cache hits and cache misses, and said set of architectural registers; and

splitting a single program into first and second concurrently executing portions which each concurrently execute distinct subsets of parallely dispatched instructions from one or more instruction streams, said first portion of said program executed on said first portion of the architecture, said second portion of said program executed on said second portion of said architecture;

~~whereby~~ wherein said second portion of said architecture is operative to prefetch data from said main memory into said architectural registers prior to being processed by said first portion of said architecture, and ~~whereby~~ wherein said second portion of said architecture is operative to move results produced by said first portion of said architecture into main memory after they are produced by said first portion of said architecture; and

prior to when said first portion of said architecture executes a conditional branch instruction, said second portion of said architecture prefetches first and second data sets from memory into said architectural registers, said first data set being needed for use as instruction operands when said condition evaluates to true, said second data set being needed for use as instruction operands when said condition evaluates to false.

3. (Currently amended) In an embedded-DRAM (dynamic random access memory) processor comprising a plurality of DRAM arrays comprising rows and columns of random access memory cells, a set of functional units which execute a first program, and a data assembly unit which executes a second program, said second program being tightly coupled with said first program, and whereby said data assembly unit is operative to load and store a plurality of data elements from a DRAM row of a main memory comprising at least one DRAM array to or from one or more register files which each include a parallel access port, a method of intelligent caching comprising the steps of:

dispatching in parallel a first sequence of instructions and a second sequence of instructions;

executing [[a]] the first sequence of instructions on said set of functional units, said functional units operative to process data stored in said register files; and

executing [[a]] the second sequence of instructions on said data assembly unit, said data assembly unit operative to transfer data between said register files and main memory;

~~whereby~~ wherein, while said first sequence of instructions is being processed, in advance of a time when one or more particular instructions in the first sequence will need to use corresponding particular data as operands, said second sequence of instructions instructs said data assembly unit to prefetch the particular data to be used as data operands to the one or more particular instructions in the first sequence of instructions into said register files from said at least one DRAM array[[s]] via said parallel access port, and, when conditional logic in said first program makes it uncertain as to which data will next be needed by said functional units executing said first sequence of instructions, said second sequence of instructions instructs said data assembly unit to prefetch time-critical the data to be used as data operands one or more particular instructions in the to the first sequence of instructions so that irrespective of the conditional outcome in processing said first sequence of instructions, the required the data to be used as data operands to the one or more particular instructions in the first sequence of instructions will be present in said registers.

4. (Currently amended) In an embedded-DRAM (dynamic random access memory) processor comprising a plurality of DRAM arrays which comprise rows and columns of random access memory cells, a set of functional units that execute a first program, and a data assembly unit that executes a second program, said second program being tightly coupled with said first program, and ~~whereby~~ wherein said data assembly unit is operative to load and store a plurality of data elements from a DRAM row to or from one or more register files which each include a parallel access port, and a selector switch operative to include or remove a register file from the architectural register set of said functional units executing ~~said~~ a first sequence of instructions of the first program, a method of intelligent caching comprising the steps of:

dispatching in parallel said first sequence of instructions and a second sequence of instructions;

executing said first sequence of instructions on said functional units, whereby said instructions involve operands, and said operands correspond to architectural registers visible to said functional units;

executing said second sequence of instructions on said data assembly unit, ~~whereby~~ wherein said execution of said second sequence of instructions is operative to prefetch information into one or more register files which are not architectural registers visible to said functional units; and

in response to progress made in said first program, said data assembly unit executing one or more instructions which transform said one or more register files which received prefetched data into architectural register files visible to said functional units and transform current architectural register files into non-architectural register files which are inaccessible to said functional units.

5. (Original) The method according to Claim 4, further comprising the step of speculatively prefetching information needed by two or more execution paths when a conditional branch in said first instruction sequence makes it ambiguous as to which data will next be needed by said functional units.

6. (Currently amended) In an embedded-DRAM (dynamic random access memory) processor that is segmented into first portion comprising at least one functional unit and second portion comprising at least one other functional unit, a method of intelligent caching comprising:

in said first portion, executing a first program comprising instructions that manipulate architectural register operands; and

in said second portion, executing a second program tightly coupled to said first program, said second program comprising an architectural register set switch command and at least one parallel data transfer command that causes data to be transferred between a parallel-loadable register file and a row of ~~said~~ a DRAM array;

wherein said second program monitors at least one bit of information generated during execution of said first program and executes said architectural register set switch command and said parallel data transfer command in support of said first program.

7. (Currently amended) In an embedded-DRAM (dynamic random access memory) processor that is segmented into first and second portions, said first portion comprising a set of functional units and a set of architectural registers accessed thereby, said second portion comprising at least one other functional unit and a first inactive register set, said other functional

unit capable of moving data between a row of a DRAM array and said ~~other~~ first inactive register set, a method of intelligent caching comprising:

splitting a single program into first and second parallely dispatched and executed portions, said first portion of said program executed on said first portion of the architecture, said second portion of said program executed on said second portion of said architecture;

~~whereby~~ wherein said second portion of said architecture is operative to prefetch data into said first inactive register set in anticipation of data requirement ~~a~~ by said first portion of said architecture; and

prior to when said first portion of said architecture executes a conditional branch instruction, said second portion of said architecture prefetches first and second data sets from memory, said first data set being needed when said condition evaluates to true, said second data set being needed when said condition evaluates to false.

8. (Currently amended) The method of Claim 7, whereby the prefetching operation further comprises:

executing first an instruction that causes a first row of a DRAM array to be loaded into ~~a~~ said first inactive register set;

executing a second instruction that causes a second row of said DRAM array to be loaded into a second inactive register set;

checking a condition; and

in response to the checking, executing a command that causes a selected one of said first and second inactive register sets to be activated to an become an architectural register visible to said first portion of said program.

9. (Original) The method of Claim 7, whereby the prefetching operation further comprises:

executing first an instruction that causes a first row of a DRAM array to be loaded into said first inactive register set;

executing a second instruction that causes a second row of said DRAM array to be loaded into a second inactive register set;

checking a condition; and

in response to the checking, executing a command that causes said architectural register set to assume an inactive state and a selected one of said first and second inactive register sets to be activated to become architectural registers visible to said first portion.

10. (Original) The method of Claim 7, further comprising:

executing in said second portion an instruction that causes data to be moved between selected registers in said first inactive register files in anticipation of a subsequent need within said first portion.

11. (Currently amended) In an embedded-DRAM (dynamic random access memory) processor comprising at least one DRAM array that comprises rows and columns of random access memory cells, at least one functional unit which executes a first program, and a data assembly unit which executes a second program, said second program being tightly coupled with said first program, and whereby said data assembly unit is operative to cause a plurality of data elements to be transferred between a DRAM row and one or more register files that each include a parallel access port, a method of intelligent caching comprising:

dispatching in parallel a first sequence of instructions and a second sequence of instructions for parallel execution;

executing [[a]] the first sequence of instructions on said at least one functional unit, said at least one functional unit operative to process data stored in said at least one register file; and

executing [[a]] the second sequence of instructions on said data assembly unit, said data assembly unit operative cause data to be transferred data between said at least one register file and said DRAM array;

whereby said second sequence of instructions instructs said data assembly unit to speculatively prefetch data in parallel from said DRAM array in support of said first sequence of instructions.

12. (Currently amended) In an embedded-DRAM (dynamic random access memory) processor comprising a plurality of DRAM arrays which comprise rows and columns of random access memory cells, at least one functional unit that executes a first sequence of instructions, and a data assembly unit that executes a second sequence of instructions, said second sequence of instructions being tightly coupled with said first sequence of instructions, and whereby said data assembly unit is operative to cause a plurality of data elements to be transferred in parallel

between a DRAM row and one or more register files via a parallel access port in each of said register files, a method of intelligent caching comprising:

executing said first sequence of instructions on said at least one functional unit, whereby said first sequence of instructions involve operands, and said operands correspond to architectural registers visible to said at least one functional unit;

executing said second sequence of instructions on said data assembly unit, whereby said execution of said second sequence of instructions is operative to cause information to be prefetched into one or more of said one or more register files; and

executing one or more instructions which transform at least one of said one or more register files into an architectural register file visible to said at least one functional unit.

13. (Original) The method according to Claim 12, further comprising:

speculatively prefetching information needed by two or more execution paths when a conditional dependency in said first instruction sequence makes it ambiguous as to which data will next be needed by said functional units.

14. (Currently amended) An intelligent-cache based embedded-DRAM (dynamic random access memory) processor comprising:

a DRAM array comprising a plurality of random access memory cells;

at least one functional unit;

at least one data assembly unit;

~~whereby~~ wherein said at least one functional unit executes a first program using instructions, and said data assembly unit executes an intelligent caching program that causes at least one row of said DRAM array to be speculatively precharged in support of the first program.

15. (Original) The intelligent-cache based embedded-DRAM processor of claim 14, further comprising:

first and second dual-port registers files, whereby the first port of each of said register files is a parallel access port and coupled in parallel to said DRAM array, and the second port of each respective register file is coupled said functional unit; and

said at least one functional unit is switchably coupled to said register files and said at least one functional unit executes at least one command to operate on one or more

architectural register operands that map onto registers within at least one of said register files.

16. (Original) The intelligent-cache based embedded-DRAM processor of claim 15, whereby the data assembly unit is responsive to an instruction set that comprises a command to transfer data in parallel between a row of said DRAM array and a selected one of said register files.

17. (Original) The intelligent-cache based embedded-DRAM processor of claim 16, whereby said command to transfer data in parallel between a row of said DRAM array and a selected one of said register files transfers data to or from said speculatively precharged DRAM row.

18. (Original) The intelligent-cache based embedded-DRAM processor of claim 17, whereby said command to transfer data in parallel between a row of said DRAM array and a selected data register file is executed to speculatively prefetch a said DRAM row into a selected one of said register files.

19. (Currently amended) An intelligent-cache based embedded-DRAM (dynamic random access memory) processor comprising:

a DRAM array comprising a plurality of random access memory cells;

at least one functional unit;

at least one data assembly unit; and

first and second dual-port registers files, whereby the first port of each of said register files is a parallel access port and is coupled in parallel to said DRAM array, and the second port of each respective register file is coupled said functional unit;

~~whereby~~ wherein said at least one functional unit executes a first program using instructions involving register operands, and said data assembly unit executes an intelligent caching program that causes at least one row of said DRAM array to be speculatively prefetched into a selected one of said register files in support of the first program.

20. (Currently amended) The intelligent-cache based embedded-DRAM processor of claim 19, whereby

each of said register files is capable of being placed into an active state and an inactive state;

said functional unit is responsive to commands involving architectural register operands that map onto ~~to~~ the registers within a register file that is in the active state; and

said selected one of said register files is in the inactive state.

21. (Currently amended) An intelligent-cache based embedded-DRAM (dynamic random access memory) processor comprising:

a DRAM array comprising a plurality of random access memory cells;

at least one functional unit;

at least one data assembly unit;

first and second dual-port registers files, whereby the first port of each of said register files is a parallel access port and is parallel coupled to said DRAM array, and the second port of each respective register file is coupled to said functional unit;

~~whereby~~ wherein said at least one functional unit is configured to execute a first program using instructions involving register operands, and said data assembly unit is configured to execute an intelligent caching program that causes data to be moved between the register files and the DRAM array in support of the first program.

22. (Original) The intelligent-cache based embedded-DRAM processor of claim 21, whereby the at least one functional unit executes instructions that exclusively include register operands.

23. (Original) The intelligent-cache based embedded-DRAM processor of claim 21, whereby the data assembly unit is responsive to an instruction set that comprises a command to perform the parallel transfer data between a row of said DRAM array and a selected data register file.

24. (Original) The intelligent-cache based embedded-DRAM processor of claim 21, whereby the data assembly further comprises:

a bit mask to select one or more data locations within at least one of said register sets; and

an instruction set which comprises a command to load a set of selected elements of a row said DRAM array into a selected set of data registers, said selection based on at least one bit in said bit mask.

25. (Currently amended) The intelligent-cache based embedded-DRAM processor of claim 21, whereby the data assembly unit further comprises:

a row address register; and

an instruction set which comprises a command to perform arithmetic on said row address register.

26. (Original) The intelligent-cache based embedded-DRAM processor of claim 25, whereby said instruction set further comprises:

a command to precharge (activate) a row pointed to by said row address register.

27. (Original) The intelligent-cache based embedded-DRAM processor according to Claim 21, further comprising:

a mask and switch unit interposed between said DRAM array and at least one of said functional units.

28. (Currently amended) The intelligent-cache based embedded-DRAM processor according to Claim 21, further comprising:

at least one coupling between said at least one ~~of said~~ functional ~~units~~ unit and said at least one ~~of said~~ data assembly ~~units~~ unit;

~~whereby~~ wherein information passed across said coupling is used to allow said at least one data assembly unit to track the execution of said first program and execute load and store operations in support thereof.

29. (Currently amended) An intelligent-cache based embedded-DRAM (dynamic random access memory) processor comprising:

a DRAM array comprising a plurality of random access memory cells;

first and second dual-port registers files, whereby the first port of each of said register files is a parallel access port and is ~~parallel~~ coupled in parallel to said DRAM array, each of said register ~~files~~ files is capable of being placed into an active state and an inactive state;

at least one functional unit that executes a first program, said functional unit coupled to said second port of said register files, said functional unit responsive to commands involving architectural register operands that map onto ~~to~~ the registers within a register file that is in the active state;

a data assembly unit that executes an intelligent caching program in support of said first program, said data assembly unit responsive to at least one command that causes data to be moved between the DRAM array and a register file that is in the inactive state;

~~whereby~~ wherein the first and second register files are capable of toggling between said active and inactive states, under program control, during program execution.

30. (Currently amended) The intelligent-cache based embedded-DRAM processor according to Claim 29, further comprising:

a control coupling between said data assembly unit and said register files, whereby said data assembly unit executes ~~comprises~~ an instruction set which comprises a command to cause at least one of the register files to toggle between said active and said inactive state.

31. (Currently amended) The intelligent-cache based embedded-DRAM processor according to Claim 29, further comprising:

a control coupling between said functional unit and said register files, whereby said functional unit executes ~~comprises~~ an instruction set which comprises a command to cause at least one of the register files to toggle between said active and said inactive state.

32. (Currently amended) The intelligent-cache based embedded-DRAM processor according to Claim 29, whereby:

said functional unit is responsive to an instruction set, and instructions within said instruction set comprise commands exclusively responsive to register operands, said register operands corresponding to a set of architectural registers; and

said data assembly unit is responsive to an instruction set, and instructions within said instruction set comprise:

(i) a command to parallel load at least a portion of an inactive register set from said DRAM array;

(ii) a command to toggle said inactive register set into said active state and at the same time to toggle an active register set into said inactive state;

~~whereby~~ wherein said architectural register set of said functional unit is dependent on the execution of said toggle command.

33. (Original) In an intelligent-cache based embedded-DRAM (dynamic random access memory) processor comprising at least one DRAM array comprising rows and columns of random access memory cells, at least one functional unit that executes a first program, and a data

assembly unit that executes an intelligent caching program in support of said first program, a method of intelligent caching processing comprising:

in said data assembly unit, causing a parallel-loadable register file to be speculatively loaded in parallel from a DRAM row;

in at least one functional unit, generating a conditional output; and

based on said condition, conditionally mapping said parallel-loadable register file to a set of architectural registers visible to said at least one functional unit.

34. (Original) The method according to Claim 33, whereby said conditionally mapping is initiated under control of said data assembly unit.

35. (Original) The method according to Claim 33, whereby said speculative loading is further controlled by a bit mask that identifies a selected subset of said register file to be speculatively loaded.

36~~35~~. (Currently amended) The method according to Claim 33, ~~whereby~~ wherein said speculative loading is further processed ~~via~~ using a mask and switch unit whereby a selected subset of said register file is speculatively loaded with a data element order permutation.

37~~36~~. (Currently amended) The method according to Claim 33, further including the steps of:

~~in~~ utilizing said data assembly unit, causing a DRAM row to be speculatively precharged in support of said first program;

~~in~~ utilizing said at least one functional unit, executing a command in said first program that causes a register value to be modified; and

~~and~~ storing at least said modified register value into said speculatively precharged DRAM row.

38~~37~~. (Currently amended) The method according to Claim 33, further including the steps of:

~~in~~ utilizing said data assembly unit, causing a DRAM row to be speculatively precharged prior to the execution of at least one event that is to be executed in said first program;

~~in~~ utilizing said at least one functional unit, executing a command to write [[a]] at least one word to at least one register in a second register set; and

~~in~~ utilizing said data assembly unit, causing at least a portion of said second register set to be written into said precharged DRAM row.

3938. (Currently amended) The method according to Claim 33, further comprising:

~~in~~ utilizing said at least one functional unit, writing an output to a register in an architectural register file;

~~in~~ utilizing said data assembly unit, reading said output from said register and using said output as a control input in said intelligent caching program.

4039. (Currently amended) In an intelligent-cache based embedded-DRAM (dynamic random access memory) processor comprising at least one DRAM array comprising rows and columns of random access memory cells, at least first and second parallel-loadable register files, at least one functional unit that executes a first program and interacts with a set of architectural register locations, and a data assembly unit that executes an intelligent caching program in support of said first program, a method of intelligent caching processing comprising:

in said at least one functional unit, executing a first group of instructions, at least some of which have operands that correspond to said architectural register locations, said architectural register locations being mapped to said first parallel-loadable register file;

in said data assembly unit, monitoring at least a subset of bits generated by the execution of said first group of instructions, and in response thereto, causing said second parallel-load register file to be parallel loaded from a row of said DRAM array, while said parallel-loadable register file is an inactive state;

in said data assembly unit, causing said second parallel-loadable register file to be mapped from said inactive state to said set of architectural register locations accessible by said at least one functional unit; and

in said at least one functional unit, executing a second group of instructions at least some of which have operands that correspond said architectural register locations;

~~whereby~~ wherein said second group of instruction cause data in said second parallel-loadable register file to be accessed.

41. (New) The embedded-DRAM (dynamic random access memory) processor of Claim 1, wherein concurrently with the execution of the first program, said second program executes said commands to unload and load at least some of said second subset of registers,

rearrange at least some data in the registers using register-to-register move commands, and to subsequently cause the second subset to become architectural registers visible to said first portion and available for processing by the first program.

42. (New) The method of Claim 2, wherein speculative prefetching of data is performed from said main memory so that the first program need not wait for the first or the second data set to be fetched from said main memory, irrespective of the outcome of the conditional instruction.

43. (New) The method of Claim 3, whereby waiting for the particular data to be loaded is avoided by having the particular data available in the register file and accessible by the first program at the time the particular instructions are executed.